oRV1 - RISC-V Architecture

This course covers and explains the implementation of the RISC-V CPU

Objective

- Understand the basics of the RISC-V architecture and instruction set.
- Develop proficiency in RISC-V C Programming and be able to write, compile, and run RISC-V C code.
- Learn how to handle interrupts and exceptions in RISC-V.
- Understand the concepts of RISC-V hardware and system design, specifically on FPGA and embedded systems

A more detailed course description is available on request at training@ac6-training.com

Course environment

- Theoretical course
  - PDF course material (in English)
  - Course dispensed using the Teams video-conferencing system
  - The trainer answers trainees' questions during the lectures and provide technical and pedagogical assistance through the Teams video-conferencing system
- Practical activities
  - Practical activities represent from 40% to 50% of course duration
  - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
  - Course will be using QEMU as the RISC-V emulator.
  - Students will be given access to a shared filesystem to save and share their work.
  - Hands-on labs will include exercises that use QEMU to emulate RISC-V systems and run assembly and C code.
  - The course will provide materials and tutorials for the lab sessions which will be based on QEMU
  - Example code, labs and solutions
- Virtual Machine with System Workbench for STM32 with GNU ARM Eclipse QEMU

Prerequisites

- Familiarity with computer architecture
- Programming skills: Some programming experience, particularly in C or assembly
- Knowledge of digital logic: Understanding of digital logic and basic concepts of computer design would be beneficial for understanding RISC-V CPU implementation and FPGA design
- Basic understanding of operating systems: Familiarity with operating system concepts such as process management, memory management, and interrupts
- The course may use Linux-based development tools and environments

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Session

Introduction to RISC-V

- Overview of RISC-V
  - What is RISC-V and why is it important?
  - History and development of RISC-V
  - RISC-V architecture and instruction set
  - RISC-V implementations and applications
- RISC-V ISA Overview
  - Instruction format
  - Instruction set encoding
  - Privileged architecture
  - Vector instructions
  - Compressed instructions

Exercise: Setting up a RISC-V development environment and running a "Hello World" program on a RISC-V emulator

RISC-V CPU Implementation

- RISC-V 32-bit CPU implementation
  - RISC-V 32-bit instruction set
  - RISC-V 32-bit register set
  - RISC-V 32-bit pipeline
- RISC-V 64-bit CPU implementation
  - RISC-V 64-bit instruction set
  - RISC-V 64-bit register set
  - RISC-V 64-bit pipeline

Exercise: CPU Implementation

Second Session

RISC-V Memory Management

- Introduction to RISC-V memory management
  - Memory management unit
  - Virtual memory
  - Address translation
- Memory-mapped I/O in RISC-V
  - MMIO interface
  - Device drivers
- Virtual memory and address translation in RISC-V
  - Page tables
  - Translation lookaside buffer

RISC-V Interrupt and Exception Handling

- Introduction to RISC-V interrupts and exception handling
  - Interrupts and exceptions
  - Interrupt handling
- Implementing interrupt handlers in RISC-V assembly and C
  - Interrupt service routines
  - Exception handling

- Handling exceptions and errors in RISC-V
  - Exception vectors
  - Trap handling

Exercise: Interrupt and Exception Handling

RISC-V C Programming and Debugging

- Introduction to RISC-V C programming
  - Setting up the C development environment
  - Writing and compiling RISC-V C code
- Debugging and testing C code
  - GDB and OpenOCD
  - Trace

Exercise: C programming and debugging

Third Session

RISC-V Optimization

- Introduction to RISC-V performance optimization
  - Understanding performance metrics
  - Identifying performance bottlenecks
- Profiling and benchmarking RISC-V code
  - Using performance counters
  - Analyzing performance data
- Optimizing RISC-V code for performance
  - Instruction scheduling
  - Loop optimization
  - Register allocation
  - Memory optimization

Exercise: Optimizing RISC-V Code

RISC-V Optimization for FPGA and Embedded Systems

- Introduction to RISC-V on FPGA
  - Overview of FPGA technology
  - RISC-V on FPGA: benefits and challenges
- Synthesis and Implementation
  - Synthesis flow
  - Place and route
  - Power and performance optimization
- Designing RISC-V systems with FPGA
  - SoC design
  - Peripherals and interfaces
  - Interrupts and exception handling
- RISC-V on embedded systems and IoT applications
  - Applications and use-cases
  - Memory and power constraints
  - Security and privacy concerns

Exercise: Implementing a RISC-V system on an FPGA development board